НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-методичний комплекс
«Інститут післядипломної освіти»

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
"___" _____ 202___ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системне програмування»

спеціальності 123 «Комп'терна інженерія»

з напряму підготовки 6.050102 «Комп'ютерна інженерія»
(код і назва)

на тему: Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання

Виконала: студенька IV курсу, групи ЗКІ-зп71
(шифр групи)

Коняхина Юлія Сергіївна
(прізвище, ім'я, по батькові) (підпис)

Керівник доц.каф.СПСКС, к.т.н. Клятченко Я.М.__
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю Клятченко Ярослав Михайлович
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент ст.викл.каф.ПМА Мальчиков В.В.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань
Слухачка _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-методичний комплекс
«Інститут післядипломної освіти»

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп'ютерна інженерія»
Освітньо-професійна програма «Системне програмування»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис)
"___" _____ 20___ р.

ЗАВДАННЯ
на дипломний проєкт слухачу
Коняхиній Юлії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема проекту: Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання
затверджені наказом НМК „ІПО" КПІ імені Ігоря Сікорського від «17»_квітня 2020 р. № 1022-с .

2. Термін подання слухачем проекту: дивитися технічне завдання

3. Вихідні дані до проекту: титульна сторінка, програма, технічне завдання (4 сторінки), пояснювальна записка (60 сторінки)

4. Зміст пояснювальної записки: перелік скорочень, умовних позначень та термінів, вступ, аналіз існуючих програмних засобів та обгрунтування теми дипломного проекту, аналіз технологій для розробки розподілених систем моніторингу, структура системи та опис роботи модулів, тестування системи та аналіз результатів, висновок, список використаної літератури.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): презентація, структурні схеми, розташування модулів програми, загальна схема проекту, схема алгоритмів, основний маршрут користувача, система моніторингу.

6. Консультанти розділів проекту[1]*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	24.04.2020	
2.	Розроблення та узгодження технічного завдання	30.04.2020	
3.	Аналіз існуючих рішень	08.05.2020	
4.	Підготовка матеріалів першого розділу дипломного проекту	15.05.2020	
5.	Підготовка матеріалів другого розділу дипломного проекту	22.05.2020	
6.	Підготовка графічної частини дипломного проекту	29.05.2020	
7.	Оформлення документації дипломного проекту	05.06.2020	
8.	Попередній огляд	12.06.2020	

Слухач _____ Юлія КОНЯХИНА
(підпис) (ініціали, прізвище)

Керівник проекту _____ Ярослав КЛЯТЧЕНКО
(підпис) (ініціали, прізвище)

[1]* Консультантом не може бути зазначено керівника дипломного проекту

Анотація

до бакалаврської дипломної роботи Коняхиної Юлії Сергіївни.

на тему : «Розробка віртуальної лабораторії функціонально-логічного моделювання»

Дипломна робота присвячена розробці зручної системи, що надає користувачеві можливість комфортно працювати зі стрічками RSS новин.

Програмний додаток надає користувачу такі можливості :

- широкий спектр методів для додавання стрічок до колекції;
- сортування колекції стрічок за допомогою зручної системи папок і фільтрів;
- модифікація та видалення стрічок з колекції;
- зручний перегляд стрічок;
- можливість працювати в декількох вікнах одночасно;
- гнучкий користувацький інтерфейс з широким набором можливостей;
- можливість поділитися цікавою інформацією з друзями через різні ;соціальні мережі;
- збереження даних користувача по закриттю програмного засобу в папці встановлення;

Користування бібліотеками, базованими на SWT в поєднанні з jRSS дозволило зробити гнучкий програмний код та зручний графічний інтерфейс користувача.

Ключові слова: крос-платформні програмні засоби , бібліотеки, базовані на SWT, бібліотека jRSS, RSS стрічки, графічний інтерфейс.

Annotation

The object is to develop cross-platform software tools for searching and organizing feeds RSS.

Purpose - Development of user-friendly system that provides the user the ability to work comfortably with the RSS news feeds.

The software application provides the user with the following features:

- A wide range of methods for adding feed in the collection;
- Arrange collection of tapes with a convenient system of folders and filters;
- Updating and deleting rows from the collection;
- Easy viewing feeds;
- The opportunity to work in multiple windows simultaneously;
- A flexible user interface with a wide range of opportunities;
- The opportunity to share interesting information with friends via various social networks;
- Saving the user data on the closure program in the installation folder ;

The use of libraries, based on the SWT in conjunction with jRSS allowed to make a flexible program code and user-friendly graphical user interface.

Key words: cross-platform software tools, libraries, based on SWT, library jRSS, RSS feeds, a graphical interface.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.467100.002 ТЗ	Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання Технічне завдання	4		
	А4	ІАЛЦ.467100.003 ТП	Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання Відомість технічного проекту	1		
	А4	ІАЛЦ.467100.004 ПЗ	Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання Пояснювальна записка	52		

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.001 ОА		
Розробив		Коняхина Ю.С.			Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Клятченко Я.М.					
Н. контроль		Клятченко Я.М.			***Опис альбому***	КПІ ім. Ігоря Сікорського, ЗКІ-зп71	
Затвердив		Романкевич В.А.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.467100.005 Д1	Розподілені програмні	1		
			Розташування модулів			
			програм			
			Схема структурна			
	А4	ІАЛЦ.467100.006 Д2	Розподілені програмні	1		
			засоби.			
			Загальна схема проекту			
			Схема структурна			
	А4	ІАЛЦ.467100.007 Д3	Розподілені програмні	1		
			засоби.			
			Схема алгоритму			
	А4	ІАЛЦ.467100.008 Д4	Розподілені програмні	1		
			засоби.			
			Система моніторингу			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.001 ОА	Арк. 3

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання».

Галузь застосування: Web-додатки.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є розробка зручної системи, що надає користувачеві можливість комфортно працювати зі стрічками RSS новин.

4. ДЖЕРЕЛО РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	24.04.2020
2.	Розроблення та узгодження технічного завдання	30.04.2020
3.	Аналіз існуючих рішень	08.05.2020
4.	Підготовка матеріалів першого розділу дипломного проекту	15.05.2020
5.	Підготовка матеріалів другого розділу дипломного проекту	22.05.2020
6.	Підготовка графічної частини дипломного проекту	29.05.2020
7.	Оформлення документації дипломного проекту	05.06.2020
8.	Попередній огляд	12.06.2020

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.467100.002 ТЗ	Програмне забезпечення	4		
			віртуальної лабораторії			
			функціонально-логічного			
			моделювання			
			Технічне завдання			
	А4	ІАЛЦ.467100.003 ТП	Програмне забезпечення	1		
			віртуальної лабораторії			
			функціонально-логічного			
			моделювання			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.467100.004 ПЗ	Програмне забезпечення	52		
			віртуальної лабораторії			
			функціонально-логічного			
			моделювання			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.003 ТП		
Розробив		Коняхина Ю.С.			Програмне забезпечення віртуальної лабораторії функціонально-логічного моделювання ***Відомість технічного проекту***	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Клятченко Я.М,					
Н. контроль		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ЗКІ-зп71	
Затвердив		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.4671.005 Д1	Розподілені програмні	1		
			Розташування модулів			
			програм			
			Схема структурна			
	А4	ІАЛЦ.4671.006 Д2	Розподілені програмні	1		
			засоби.			
			Загальна схема проекту			
			Схема структурна			
	А4	ІАЛЦ.4671.007 Д3	Розподілені програмні	1		
			засоби.			
			Схема алгоритму			
	А4	ІАЛЦ.4671.008 Д4	Розподілені програмні	1		
			засоби.			
			Система моніторингу			

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.003 ТП	Арк. 3

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ , СКОРОЧЕНЬ І ТЕРМІНІВ

САПР – система автоматизованого проектування

HTTP – HyperText Transfer Protocol — «протокол передачи гипертекста»

JSP – Java Server Pages

AJAX – Asynchronous JavaScript And XML

XML – Extensible Markup Language

DOM – Document Object Model

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

DHTML – Dynamic HTML

ВСТУП

Ми живемо у часи постійного та активного розвитку інформаційних технологій. Активно створюються веб-додатки, основою яких виступають найрізноманітніші розробки. За рахунок цього доступ до таких розробок значно полегшується, а також налагоджується командна робота з ними. Зручно та просто мати доступ до різноманітних речей лише за допомогою встановлення браузера. Якщо розмістити певні дані на сервері, це дасть змогу працювати над певною розробкою або проектом дистанційно та навіть з різних пристроїв, звичайно, за умови їх синхронізації.

Більшість розробників програмного забезпечення регулярно потрапляють у ситуацію, коли потрібно запустити або швидко перевірити свій код, але далеко не всі знають, що для вирішення даної проблеми немає необхідності запускати інтегровані середовища розробки (IDE). Достатньо просто використати певні онлайн-інструменти, які значно прискорюють цей процес. Якщо розробити віртуальну лабораторію функціонально-логічного моделювання, це дасть змогу проводити Verilog симуляції дистанційно та без встановлення додаткових модулів на власний електронний пристрій. Однак у подібних сервісів є певні перешкоди, однією з яких є збільшення навантаження на систему під час їх використання. Однак, якщо зменшити функціонал таких сервісів, вони можуть знайти своє місце на ринку програмного забезпечення. Крім того, завдяки вищезгаданому розвитку технологій та збільшенням обчислювальної потужності електронних пристроїв, ця ідея розробки безпосередньо у веб-браузері може бути досить перспективною.

Такі віртуальні лабораторії надають змогу розробникам отримати можливість провести симуляцію на мові опису апаратури Verilog, при чому

для цього вони потребують лише пристрій із встановленим на ньому веб-браузером.

Метою даної роботи є дослідження існуючі лабораторії функціонально-логічного моделювання та реалізація власної, а також аналіз отриманої реалізації.

1. ПРИНЦИПИ ПОБУДОВИ ВЕБДОДАТКІВ ДЛЯ СИСТЕМ АВТОМАТИЧНОГО ПРОЕКТУВАННЯ

1.1 Принципи побудови веб-додатків

Наразі використовуються різноманітні типи технологій створення Web-додатків. Мета у таких додатків є спільна - реалізація бізнес-логіки на стороні сервера, а також генерація коду для клієнтів. Крім того, вони мають таку саму архітектуру взаємодії клієнта та сервера, загальний протокол взаємодії HTTP. Загалом логіку функціонування клієнт-серверних програм можна побачити на малюнку 1.

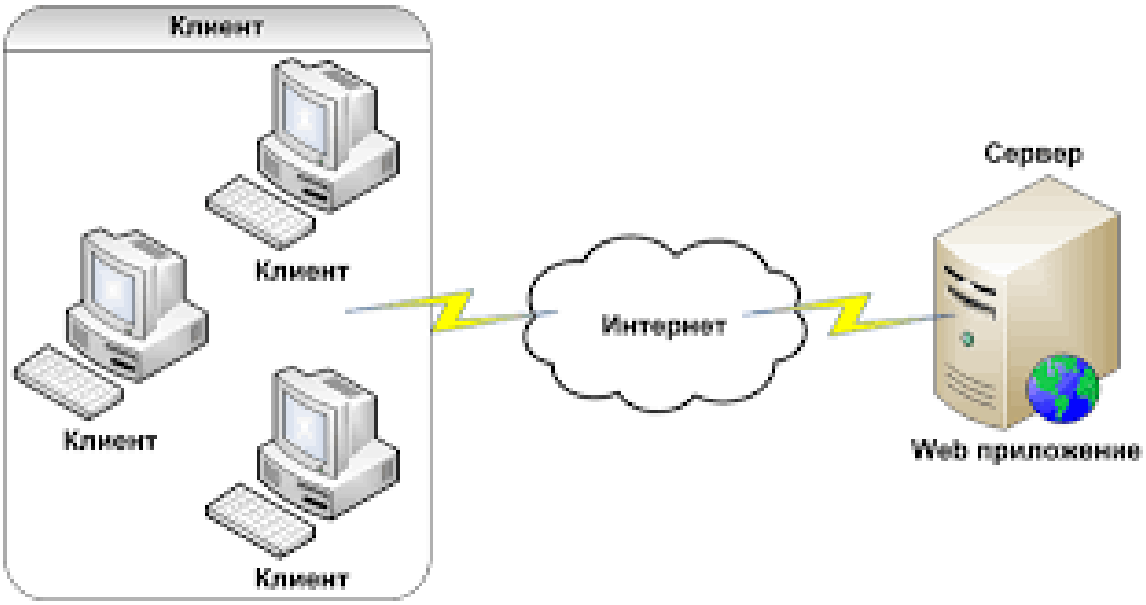

Рисунок 1 - Архітектура роботи клієнт-серверних додатків

серверної стороні для подолання цих складностей. Нижче розглянуто різноманітні підходи до створення клієнт-серверних додатків, їх сильні та слабкі сторони, а також конкретні платформи.

Принципи клієнт-серверних додатків

У разі розглядання платформ для створення додатків потрібно виділити два основних принципи:

1. Обробка запитів та формування відповідей.
2. Вбудовування програмного коду в шаблони HTML сторінок.

Перший підхід дозволяє найкращим чином управляти обробкою та підвищувати продуктивність. Для цього передаються всі дані про запит коду, який безпосередньо виконується, а цей код може, в свою чергу, як сформувати відповідь зі сторінкою для користувача, так і відкрити потік двійкових даних на передачу. Але при цьому всі дані для передачі будуть сформовані програмним шляхом, за рахунок чого розробка звичайних сторінок буде уповільнена, а також ускладнюється кооперація між верстальником та програмістом. Як приклад, можна навести технології CGI, Java Servlets.

Якщо говорити про другий підхід, то можна сказати, що у ньому використовуються шаблони сторінок користувача. Вони мають бути оформлені так, щоб у них можна було вставити фрагменти програмного коду. Це вельми ефективно у разі створення простих додатків, які відрізняються статичністю основної інформації, при цьому динамічна інформація генерується простими програмними конструкціями. Якщо мова йде про розробку складних програмних систем, взаємодія між компонентами та реалізація складної архітектури значно

Можна побачити, що робота додатків відбувається наступним чином:

1. Спочатку клієнт за допомогою веб-браузеру ініціює запит до сервера.

2. Після цього дані запит обробляється та готується відповідь. Веб-сервер обробляє ресурс, щодо якого надійшов запит. Якщо потрібен статичний ресурс, наприклад, малюнок, документ, ця інформація буде відформатована для протоколу HTTP ті передана клієнтові як відповідь на його запит. Якщо був запит щодо динамічного ресурсу, тоді він (запит) передається на обробку певного контейнеру веб-додатків, де і відбуватиметься подальша робота.

3. У будь-якому разі дані будуть передані клієнту після форматування для протоколу HTTP. У відповіді будуть включені дані (двійкові або у вигляді HTML коду), а також необхідні додаткові параметри, які будуть передані у заголовку HTTP відповіді.

Наведений сценарій є однаковий для всіх додатків з боку сервера. Зрозуміло, що з подібним підходом виникають проблеми при створенні веб-додатків, у основі яких лежить відсутність стану у веб-додатка (так зване stateless programming), тобто додаток функціонує лише у режимі запит-відповідь, при цьому він не має даних про попередні дії користувача або будь-якої іншої постійної інформація. Для того, щоб уникнути таку проблему, можно використати так звану користувальницьку сесію, за допомогою якої можна зберігати дані на сервері під час сеансу роботи користувача.

Однак одна лише сесія не може вирішити усі складності, які виникають при створенні web-додатків.

ускладнюється. Крім того, продуктивність також знижується, що може призвести до обмеження можливостей реалізації складних сторінок. Приклад таких технологій - PHP, ASP, JSP.

Окрім того, платформи розробки на різному рівні задовольняють вимогам сьогодення, які виникають при побудові складних Web систем. Найважливішими з таких вимог, які роблять систему зручною та ефективною в експлуатації, є наступні:

- незалежність платформи.
- мова, на якій вона була реалізована.
- масштабованість та продуктивність.
- перспективи розширення та інтеграції.
- простота і зручність використання, засоби розробки, які є у наявності.
- наявність необхідних програмних бібліотек.

Таким чином, були визначені певні вимоги, яким мають відповідати сучасні платформи розробки. Далі будуть розглянуті найпопулярніші платформи, їх специфіка, а також оцінка за наведеними принципами.

Швидкість розвитку інформаційного Web-середовища призвела до зміни вимог до Web-додатків. Наприклад, існує тенденція до створення Web-додатків, які мають такі самі можливості, які є у звичайних додатків, призначених для настільних систем. Але у випадку, коли працюють програми, які підтримують мережеву взаємодію, стає неможливим усунення затримки відповіді, яка пов'язана з передачею даних через Інтернет. Для зменшення негативного ефекту від затримки даних використовується технологія Ajax. Але ця технологія

повністю змінила структуру та принципи роботи Web-додатків. Сучасні мережеві програми все більше й більше функцій віддають стороні клієнта, збільшуючи таким чином обсяг коду клієнтської частини Web-додатку, тому робота над нею виконується групою розробників. Як результат, було виявлено, що мова JavaScript, що використовується при створенні Ajax-додатків, має досить специфічні особливості застосування і не відповідає тим вимогам, які виникають при налагодженні програм та інструментальних засобів розробки.

Можна розділити задачі, які постають перед додатком. Для цього потрібна взаємодія між JavaScript та Java. Якщо на Java написати код, який буде реалізувати складні алгоритми, можна використати велику кількість інструментальних засобів для розробки та налагодження програм. У цьому випадку на долю JavaScript залишаться невеликі за обсягом уривки коду, які будуть динамічно змінювати вміст сторінки, та можуть бути написані та налагоджені без використання спеціальних інструментальних засобів розробки та налагодження програм.

Такий підхід можна реалізувати шляхом використання набору базових засобів для написання Ajax-додатків, створивши надалі віртуальну лабораторії функціонально-логічного моделювання.

1.1.1 Типи додатків

Розширення (англ. Extension) використовуються для зміни наявних функцій, а також для надання нових можливостей. Розширення є досить популярними у браузері Firefox, оскільки його розробники створили браузер у вигляді мінімалістичної програми для запобігання росту кількості помилок та громіздкості, при цьому високий рівень розширення було збережено, щоб індивідуальні користувачі могли додавати необхідні для них функції.

Розширення технологій

- CSS (Cascading Style Sheets)
- DOM (Document Object Model) - застосовується для зміни XUL в режимі реального часу або зміни вже завантаженого HTML
- JavaScript - основна мова браузера Mozilla
- XPCOM (кросплатформлена модель компонентних об'єктів)
- XpConnect
- XPI
- XUL (XML-мова інтерфейсу користувача) - застосовується для визначення інтерфейсу користувача та взаємодії з ним.

Додаткові можливості

Як правило, розширення використовують для надання програмі нових можливостей. За рахунок розширення можна, наприклад, додати наступні можливості: читачі RSS, менеджери закладок, електронна пошта, пенали, засоби веб-розробки тощо. Також деякі розширення Firefox виконують функції, що у

попередні часи були частиною Mozilla Suite, наприклад, ChatZilla, клієнт IRC та календар.

Певні розширення мають змогу змінити вміст веб-сторінки при її появі на екрані. Як приклад, розширення Adblock запобігає завантаженню реклами.

1.1.2 Технологія Web 2.0

Завдяки появі та розвитку Web 2.0 стало можливим створити динамічні веб-додатки та віртуальні лабораторії функціонально-логічного програмування. Виділяють кілька понять - Web-служби, Ajax, Mash-up, теги і т.д.

Web-служби - програми, з якими можна взаємодіяти через протокол HTTP, а обмінюватися даними у форматі XML, JSON і подібних до них. Завдяки цьому програмне забезпечення може використовувати Web-служби замість самостійної реалізації потрібних функцій.

Ajax (Asynchronous JavaScript and XML) - це такий підхід до побудови Web-програм, за яким Web-сторінка асинхронно та без перезавантаження може отримати необхідні дані з сервера для користувача. Нерідко Ajax вважають синонімом Web 2.0, але це помилково, тому що Web 2.0 не має прив'язки до будь-яких технологій і є скоріше тенденцією розвитку Інтернету.

Mash-up - такий принцип використання функцій різних джерел даних, завдяки чому виникають нові можливості. З технічної сторони це веб-сайт, який поєднує залежні між собою сервіси, створюючи новий веб-сервіс.

Теги - релевантні слова, які надаються певним об'єктам або частинам інформації. Це своєрідні ярлики, які описують об'єкт, щоб надати йому певну

позицію між іншими об'єктами, таким чином здійснюється класифікація на основі таких ключових слів, пошук інформації. Вибір тега зазвичай індивідуальне рішення, яке залежить від того, хто створив інформаційний фрагмент. Тегів у об'єкта може бути декілька.

Також існує поняття “багатих Веб-програм” - це такі, що мають функції звичайних програм, але вони виконують свою роботу у браузері та мають активну взаємодію із сервером. Завдяки цьому робота виконується з більшою ефективністю.

Інтерфейс таких програм виглядає так само, як і інтерфейс класичних програм, а не як у web-програм, тому проблем з ефективним їх використанням не виникає навіть у користувачів з мінімальними знаннями про Інтернет.

Технологія Web 2.0 має наступні особливості:

- синдикацію
- протоколи передачі даних
- клієнтське програмне забезпечення
- браузери з плагінами й розширеннями
- CSS або cascading style sheets, каскадні таблиці стилів - це спеціальна мова стилю сторінок, яка застосовується для описання їх зовнішнього вигляду. Це є одна з основних технологій інтернету, так само, як HTML та JavaScript.

1.2 Web - технології для створення web-додатків

1.2.1. Базові технології Web.

HTML - HyperText Markup Language - це мова тегів, на якій створюються документи для мережі Інтернет. HTML є стандартною мовою розмітки документів. HTML-документи потрапляють до веб-браузерів з веб-сервера або локальної пам'яті, а ті, в свою чергу, інтерпретують мову HTML у формат, зрозумілий та зручний для користувача.

Цю мову було створено на початку 90 х років, як мову, що призначалася для обміну науковою та технічною документацією. При чому вона була зручною та зрозумілою для людей, які не розуміються на верстці. Завдяки визначенню невеликого об'єму структурних та семантичних елементів, вона зменшує проблеми зі складністю SGML, що спрощує структуру документів. Також HTML підтримує гіпертекст. З часом було додано і мультимедійні можливості.

Для форматування текстових документів із кодом HTML використовуються веб-браузери - спеціальні програми.

Як було вказано вище, HTML створювався для форматування документів без прив’язки до засобів відображення. Але зараз HTML використовується для інших задач, наприклад, було додано засоби для використання мульдимедіа, формування графічних оформлень, можливості підключення плагінів та розширень.

Для створення динамічних сторінок розробили велику кількість нових технологій, таких як JavaScript, Java applets, Adobe Flash, Microsoft Silverlight. Деякі з них було інтегровано в браузери (Javascript), інші потребують підключення певних плагінів, але на сайтах розробників їх можна отримати

безкоштовно, якщо їх не було доставлено разом з операційними системами чи браузерами.

1.2.2 Загальні відомості про Ajax

Asynchronous JavaScript And XML - спосіб створення інтерфейсу користувача, при якому сторінка обмінюється даними з Web-сервером у фоновому режимі, довантажуючи при цьому самостійно необхідні дані. Сама сторінка при цьому не перезавантажується, що дозволяє програмі працювати швидше та зручніше. Це підхід до використання декількох близьких технологій, він базується на принципах використання технології взаємодії із сервером за допомогою JavaScript об'єкта XMLHttpRequest, використання DHTML для динамічної зміни вмісту сторінки, динамічного створення дочірніх фреймів. Завдяки цим прийомам можна створити більш зручний веб інтерфейс користувача там, де необхідна активна з ним взаємодія. Для того, щоб дані від сервера до клієнта було передано, використовуються формати XML або JSON.

Ajax є асинхронним, завдяки чому користувач має можливість переглядати сайт далі, пока сервер продовжує обробляти запит. Ajax став популярним з часів, коли Google почала активно використовувати його для Google maps, Gmail та Google Suggest. Коли вони були створені, ефективність Ajax була підтверджена.

У програмах, які створювалися з використанням Ajax, певні функції переходять з сервера на клієнта. Така програма реагує самостійно на певні процеси з боку користувача. Завдяки тому, що HTML документ є у розпорядженні клієнта під час роботи з програмою, він може зберігати всю інформацію про її стан.

Концепція динамічного завантаження вмісту не є новою, так, була можливість завантажити зовнішній сценарій JavaScript, який змінював поточну сторінку. Це відбувалося за допомогою атрибуту src, при чому у зв'язку з його лімітами та додатковими навантаженнями на сервер, це було не найкраще рішення, тому що від цього методу вимагалося виконання певних додаткових дій для створення певного сценарія JavaScript, що матиме інструкцію по модифікації поточної сторінки у нову.

Більше того, засоби Ajax не являються унікальними для забезпечення асинхронного обміну даними з сервером. Як приклад, можна навести Macromedia Flash (4 версія та пізніше), яка вміє завантажувати дані у форматі XML або CSV з серверу, при цьому сторінка при цьому не перезавантажується. Але Macromedia не може бути використана для побудови багатих веб-програм, тому що її зазвичай використовують для опрацювання мультимедійних даних і вона не є придатною для динамічної зміни вмісту веб-сторінок.

До переваг Ajax можна віднести наступні:

- Формування веб-програм, які мають інтерфейс, а також багаті функції, що є подібними до звичайних програм, але завдяки активній роботі з сервером, веб-програми мають великі переваги у порівнянні зі звичайними програмами.
- Економія трафіку - усю сторінку завантажувати не потрібно, вистачає завантаження невеликої її частини, яку було змінено.
- Зменшення навантаження на сервер - сервер не повинен кожного разу згенерувати всі сторінки, достатньо тільки змінений фрагмент.
- Прискорення реакції інтерфейсу - завдяки завантаженню сегменту сторінки користувач має змогу побачити результати своєї роботи значно швидше.

Також Ajax має певні недоліки:

- Немає інтеграції зі стандартними інструментами браузера: не можна повернутися назад, згенеровану при допомозі Ajax сторінку неможливо додати до закладок.
- Пошукові роботи не можуть індексувати сайт, так як у них немає підтримки JavaScript.

1.2.3 XHTML

XHTML - це заснована на XML мова розмітки гіпертексту, що є максимально наближеною до поточних стандартів HTML. XHTML відрізняється від HTML суворістю написання коду. Якщо HTML дозволяв писати практично будь-які конструкції та браузер їх правильно розпізнював, то із появою XHTML це стало неможливо, тому що вона потребує суворого дотримання всіх правил, які вимагає W3C. Суворі вимоги до оформлення XHTML-коду дозволяють уникнути багатьох помилок ще на стадії написання та налагодження.

Для поціновувачів HTML потрібно відмітити, що XHTML - це нова мова, яка прийшла на заміну старому HTML. Нових версій HTML не буде. В результаті всі браузери мають перейти на XHTML, при чому має залишитися сумісність зі старим HTML, але не більше того. Також можна навести цитату зі специфікації XHTML від W3C: “Родина XHTML створювалася із урахуванням загальної сумісності користувальницьких агентів. За допомогою нового механізму профілювання користувальницьких агентів та документів сервери, проксі та користувальницькі агенти зможуть перетворювати зміст найкращим чином.

У кінцевому результаті стане можливою розробка відповідного XHTML змісту, який буде придатним для будь-якого відповідного XHTML користувальницького агенту”.

XHTML є сумісним з HTML за умови дотримання деяких правил, опис яких можна знайти у стандартах. Це означає, що навіть найстаріші браузери, які розуміють HTML, будуть працювати і з XHTML.

Для перевірки вірності написання XHTML-коду існують програми-валідатори. Крім того, користувальницькі агенти, які підтримують XHTML, будуть самостійно повідомляти розробників про помилки у синтаксисі, якщо такі виникнуть.

Відмінності XHTML 1.0 від HTML.

Існує кілька вимог, які розробник повинен виконувати:

- на початку документу потрібно вказати один із можливих DTD - Document Type Definition - визначення типу документу:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
        <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

- у тілі XHTML-документу повинні обов'язково бути присутніми наступні теги: html, head, title и body;
- обов'язковим є також наявність атрибуту xmlns у елементі html;
- імена тегів та атрибутів повинні записуватися у нижньому регістрі;
- усі значення атрибутів повинні записуватися у лапки;
- усі теги повинні закриватися, причому, якщо елемент не має закриваючий тег, потрібно додавати в його кінці слеш (
 або
 - пробіл для сумісності зі старими браузерами);
- необхідно зберігати правильну вложеність тегів (текст — невірно; потрібно писати текст);
- забороняється використовувати мінімізовані атрибути (nowrap потрібно замінити на nowrap="nowrap"); повний перелік таких атрибутів: checked, compact, declare, defer, disabled, ismap, noresize, noshade, nowrap, multiple, readonly, selected.
- на наступні елементи накладаються обмеження по додаванню до них інших елементів:
  - a не може містити інші елементи a;
  - form не може містити інші елементи form;
  - label не може містити інші елементи label;
  - pre не може містити, object, big, small, sub або sup;
  - button не може містити елементи input, select, textarea, label, button, form, fieldset, iframe або isindex;

- спеціальні символи у істинному значенні мають замінюватися на свої еквіваленти:
  - & на &;
  - < на <;
  - > на >.

Крім того, існує ряд необов'язкових рекомендацій, які розробники не повинні виконувати у версії XHTML 1.0, але у наступних версіях цієї мови можливо рекомендації перетворяться на вимоги:

- декларація XML-документу на самому початку коду перед DTD (<?xml version="1.0" encoding="windows-1251"?>);
- елемент title йде відразу після відкриття тегу head;
- використання атрибуту id замість name (name вважається застарілим атрибутом);
- наявність атрибуту type у елементах файлів, які підключаються (таблиць стилів та скриптів);
- відмова від використання атрибуту target.

Наведемо мінімальний код правильної XHTML - сторінки:

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ru" lang="ru">
<head>
  <title>Заголовок</title>
</head>
<body>
  Зміст документу
</body>
</html>
```

Якщо вставити цей код у файл, зберегти його як “file.html” та відкрити через веб-сервер, то вся інформація буде отримана клієнтом як >text/html<. Тобто, як і звичайна HTML-сторінка. Фактично для браузера це буде не XHTML, а HTML-документ.

У XHTML є свій власний MIME-тип: <application/xhtml+xml>.

MIME - це спеціальний набір розширень, який вказує програмам, як обробляти вхідну інформацію. Спершу MIME-типи були розроблені для поштових програм, звідки вони і отримали свою назву.

Отже, XHTML-дані потрібно правильно віддавати клієнту саме у форматі <application/xhtml+xml>, тому що всі переваги крос-браузерності (збільшення швидкості аналізу коду процесором XML, повідомлення про помилки браузером і таке інше), можуть бути досягнуті лише у тому випадку, коли користувальницький агент підтримує XHTML та йому повідомляється про те, що вхідні дані - XHTML-код.

Єдине, що потрібно пам'ятати при надсиланні XHTML-коду:

якщо браузер розуміє XHTML, то інформацію можна надіслати як <application/xhtml+xml>, якщо ж ні - то тільки як <text/html>.

Список сучасних агентів, які підтримують XHTML: MZ, Opera, Safari. Internet Explorer, на жаль, поки що не розуміє <application/xhtml+xml>. Перевірити, чи підтримує користувальницький агент необхідний MIME-тип можна по вихідному із браузера заголовку <Accept>, де містяться усі MIME-типи, які відомі клієнту. Наведемо приклад, як це можна зробити за допомогою Perl-скрипту:

```
#!/usr/bin/perl -w

# Дізнаємося, чи браузер підтримує XHTML.

my $html  = "text/html";

my $xhtml = "application/xhtml+xml";

my $type  = $ENV{HTTP_ACCEPT} =~ m/\Q$xhtml\E(?!\s*;\s*q=0)/ ? $xhtml : $html;

# Виводимо відповідний заголовок.

print "Content-Type: $type\n\n";

# Вивід (X)HTML-документу.

print "...";
```

При надсиланні даних як <application/xhtml+xml> потрібно враховувати ще декілька моментів, без яких можливим стає поява помилок. Так як з синтаксичної точки зору XHTML - це XML, елементи <script> і <style> у XHTML - це #PCDATA-блоки (а не #CDATA). Зміст таких блоків необхідно помістити у спеціальну секцію CDATA, тому що в іншому разі процесор XML перетворить спеціальні символи у їх еквіваленти ще до обробки таблиці стилів чи сценарія браузером. Наступний приклад показує, як це можна зробити:

```
<script type="text/javascript"><!--//--><![CDATA[//><!--

...

//--><!]]></script>

...

<style type="text/css"><!--/*--><![CDATA[/*><!--*/

...

/*]]>*/--></style>
```

Такий синтаксис є універсальним. Цей код буде коректно працювати як при <text/html>, так і при <application/xhtml+xml>. Гарним і найпростішим рішенням буде підключення зовнішніх файлів таблиць стилів та скриптів. У XHTML це робиться так само, як і в HTML:

```
<!-- Підключення CSS-файла (не забудьте про закриваючий слеш). -->

<link rel="stylesheet" type="text/css" href="file.css"

title="" media="screen" />


<!-- Підключення JS-файла. -->

<script type="text/javascript" src="file.js"></script>
```

1.2.4 CSS

Cascading Style Sheets або CSS, каскадні таблиці стилів - це стандарт стилів, які було оголошено консорціумом W3C. Термін “каскадні” вказує на можливість злиття різноманітних видів стилів та на наслідування стилів внутрішніми тегами від зовнішніх.

CSS - це мова, яка містить набір властивостей для визначення зовнішнього вигляду документа. Специфікація CSS визначає властивості та описувальну мову для встановлення зв'язку з HTML-елементами.

CSS - абстракція, у якій зовнішній вигляд Веб-документа визначається окремо від його змісту.

Деякі стилі підтримуються не усіма браузерами. Однак можна назначати будь-які стилі, тому що ті, що не підтримується, будуть просто проігноровані.

За методами додавання стилів до документу розрізняють три види стилів.

Внутрішні стилі

Внутрішні стилі визначаються атрибутом style конкретних тегів. Внутрішній стиль діє тільки на елементи, які було визначені такими тегами. Цей метод мало відрізняється HTML.

Приклад:

```
<p style="color:blue">Абзац із текстом синього кольору</p>
```

Абзац із текстом червоного кольору

РЕЗУЛЬТАТ:

Абзац з текстом синього кольору

Абзац з текстом червоного кольору

Не варто використовувати внутрішні стилі занадто часто, так як тоді Веб-документ буде перевантажено кодом та його зовнішній вигляд буде важко змінити.

Глобальні стилі

Глобальні стилі CSS розміщуються у контейнері <style>...</style>, які розміщено в свою чергу у контейнері <head>...</head>.

Атрибут type=”text/css”, який раніше був обов'язковий для тегу <style>, у стандарті HTML5 можна опустити.

Глобальні стилі є універсальним засобом, який дозволяє не лише оперативно змінювати зовнішній вигляд сторінки, але й боротися з перевантаженістю документа оформлюючими тегами. Проблема в тому, що такі стилі потрібно прописувати на кожній сторінці сайту.

Приклад:

<html>

<head>

.....

```
    <style type="text/css">
        p {color:#808080;}
    </style>
    .....
</head>
<body>
    <p>Сірий колір тексту в усіх абзацах Web-сторінки</p>
    <p>Сірий колір тексту в усіх абзацах Web-сторінки</p>
</body>
</html>
```

РЕЗУЛЬТАТ:

Сірий колір тексту в усіх абзацах Web-сторінки

Сірий колір тексту в усіх абзацах Web-сторінки

Зовнішні (зв'язані) стилі

Зовнішні (зв'язані) стилі визначаються у окремому файлі з розширенням css. Зовнішні стилі дозволяють усім сторінкам сайту виглядати однаково.

Для зв'язку з файлом, у якому описані стилі, використовується тег <link>, який розташований у контейнері <head>...</head>.

У цьому тезі мають бути задані два атрибути: rel="stylesheet" и href, який визначає адресу файла стилів.

RSS (Really Simple Syndication) - сімейство XML-форматів, призначення для опису стрічок новин, анонсів статей, тощо. Інформація з різних джерел, представлена в форматі RSS, може бути зібрана, оброблена и представлена користувачеві в зручному для нього вигляді спеціальними програмами - агрегаторами.

Технологія RSS змінила способи отримання мережевого контенту. За допомогою цієї технології можна транслювати практично будь-який матеріал, розділивши його на окремі частини - випуски.

Наразі в мережі можна знайти велику кількість різноманітної інформації. Важко сказати, які сфери людського життя зараз не охоплює інтернет. Але, незважаючи на широкий спектр послуг, завжди існують певні труднощі для пошуку якісного джерела інформації.

Інформація – один із найважливіших аспектів людського життя. А знаходження необхідної інформації іноді може зайняти велику кількість часу.

Причиною для створення довідкової системи стали труднощі пошуку каналу RSS новин з необхідним наповненням та труднощі при роботі з декількома каналами одночасно. Цю та інші проблеми має вирішити дана програмна розробка.

Вибір технології для створення системи

В наш час існує велика кількість технологій, які надають змогу реалізувати ідею проекту. Крос-платформенних серед них набагато менше. З найбільш поширених можна виділити дві – JAVA Standart Edition та C#.

Обидві технології мають дуже схожий синтаксис програмного коду та в обидві було закладено схожі принципи. Після проведення аналізу технологій я вирішив обрати JAVA SE. Дві основні причини мого вибору

- C# є, нажаль, лише умовно крос-платформенною, адже, фактично, .NET Framework існує лише для операційних систем сімейства Microsoft Windows.
- Відсутність можливості скористуватися повною версією Microsoft Visual Studio та незадовільність її скороченої версії всім необхідним умовам для виконання поставленої задачі.

Тепер зосередимося на можливостях обраної технології – JAVA SE. У створенні мови програмування Java було п'ять початкових цілей:

- синтаксис мови повинен бути «простим, об'єктно-орієнтовним та звичним».
- реалізація має бути «безвідмовною та безпечною», а також «високопродуктивною».
- повинна зберегтися «незалежність від архітектури та портативність».
- мова має бути «динамічною, інтерпретованою та підтримувати мультиопрацьовування».

Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Цього можна досягти, компілюючи початковий Java код у байт-код, який являє собою спрощені машинні команди. Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

У намірах проектувальників Java мала замінити C++ — об'єктного наступника мови C. Проектувальники почали з аналізу властивостей C++, які є причиною найбільшого числа помилок, щоби створити просту, безпечну і безвідмовну мову програмування.

В Java існує система винятків або ситуацій, коли програма зустрічається з неочікуваними труднощами, наприклад:

- операції над елементом масиву поза його межами або над порожнім елементом
- читання з недоступного каталогу або неправильної адреси URL
- ввід недопустимих даних користувачем

1.3 Аналіз наявних середовищ розробки програмного коду Java

Інтегроване Середовище Розробки — від Integrated Development Environment — це комп'ютерна програма, що допомагає програмістові розробляти нове програмне забезпечення чи модифікувати вже існуюче.

Інтегровані середовища розробки зазвичай складаються з редактора програмний коду, компілятора чи інтерпретатора, засобів автоматизації збірки, та зазвичай зневаджувача. Іноді сюди також входять системи контролю версій, засоби для профілювання, а також різноманітні засоби та утиліти для спрощення розробки графічного інтерфейсу користувача. Багато сучасних серидовищ також включають оглядач класів, інспектор об'єктів та діаграм ієрархії класів для використання об'єктно-орієнтованого підходу у розробці програмного забезпечення.

Розглянемо три найбільш популярні середовища розробки програмного коду на мові Java:

1. ***Eclips IDE***

Eclipse — вільне модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation. Написаний в основному на Java, і може бути використаний для розробки застосунків на Java і, за допомогою різніх плагінів, на інших мовах програмування. Випущена на умовах Eclipse Public License, Eclipse є вільним програмним забезпеченням.

Eclipse являє собою фреймворк для розробки модульних кросс-платформових застосунків із низкою особливостей: можливість розробки ПЗ на багатьох мовах програмування (рідною є Java);

- крос-платформова;

- модульна, призначена для подальшого розширення незалежним розробниками;
- з відкритим вихідним кодом;
- розробляється і підтримується фондом Eclipse, куди входять такі постачальники ПЗ, як IBM, Oracle, Borland.

GUI в Eclipse написаний з використанням інструментарію SWT. Останній, на відміну від Swing (який лише емулює окремі графічні елементи використовуваної платформи), дійсно використовує графічні компоненти даної системи. Призначений для користувача інтерфейс Eclipse також залежить від проміжного шару GUI, званого JFace, який спрощує побудову призначеного для користувача інтерфейсу, що базується на SWT.

2. ***NetBeans IDE***

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за умовчанням підтримує розробку для платформ J2SE і J2EE.

За якістю і можливостям останні версії NetBeans IDE змагаються з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше.

NetBeans Platform пропонує багаторазово використовувані сервіси і модулі для настільних додатків, дозволяючи розробникам сфокусуватися на логіці програми. Особливості платформи:

- управління дизайном програми (меню, спливаючі вікна)

- управління настройками користувача
- управління зберіганням даних
- управління вікнами
- фреймворк для розробки покрокових майстрів установки.
- NetBeans Visual Library - бібліотека візуальних елементів
- Integrated Development Tools - вбудовані інструменти розробки
- невелика кількість додаткових бібліотек (відсутня базова бібліотека роботи з поштою).

3. JDeveloper

JDeveloper - безкоштовна інтегрована середа розробки програмного забезпечення, розроблена корпорацією Oracle. Надає можливість для розробки на мовах програмування Java, XML, SQL і PL / SQL, HTML, JavaScript, BPEL і PHP. JDeveloper покриває весь життєвий цикл розробки програмного забезпечення від проектування, кодування, налагодження, оптимізації та профілювання до його розгортання.

Виробник зазначає в якості основного завдання середовища - максимальне використання можливостей візуального та декларативного підходу до розробки програмного забезпечення на додаток до зручної середовищі кодування. Oracle JDeveloper інтегрована з Oracle Application Development Framework - Java EE-каркасом для створення комерційних додатків на Java.

В кожної середи розробки є свої недоліки і переваги. Виходячи з необхідності користування декількома бібліотеками а також з певного досвіду середою для розробки було обрано Eclipse ID.

1.4 Аналіз та вибір додаткових бібліотек Java

На сьогоднішній день існує безліч Java бібліотек, отже необхідно вирішити які бібліотеки можуть допомогти нам в розробці програмного додатку.

Так як в планах розробка дійсно зручного інтерфейсу користувача, консольного режиму буде недостатньо. Отже необхідно обрати певну бібліотеку для розробки нашого інтерфейсу. Java пропонує ряд таких бібліотек – SWT, JWT, Swing. Standard Widget Toolkit, або SWT (вимовляється «світ») - бібліотека з відкритим вихідним кодом для розробки графічних інтерфейсів користувача на мові Java.

Розроблено фондом Eclipse, ліцензується під Eclipse Public License, одній з ліцензій відкритого ПЗ.

SWT не є самостійною графічною бібліотекою, а являє собою кросс-платформенну оболонку для графічних бібліотек конкретних платформ, наприклад, під Linux SWT використовує бібліотеку Gtk +. SWT написана на стандартній Java і дістає доступ до OS-специфічних бібліотек через Java Native Interface.

SWT - альтернатива AWT і Swing (Sun Microsystems) для розробників, які бажають отримати звичний зовнішній вигляд програми в даній операційній системі. Використання SWT робить Java-додаток ефективнішим, але знижує незалежність від операційної системи та обладнання, вимагає ручного звільнення ресурсів і в деякій мірі порушує Sun-концепцію платформи Java.

Для написання користувацького інтерфейсу я обрав саме SWT, бо вважаю, що він надає всі необхідні можливості для написання зручного багатофункціонального кросс-платформенного інтерфейсу користувача Тим паче, що яскравим прикладом використання даної бібліотеки є вже обрана

нами середа розробки, Eclipse IDE, графічний інтерфейс користувача якої базується саме на SWT.

Також я вважаю за доцільне обрати певну бібліотек для роботи з RSS файлами. Таких бібліотек не так вже й багато, тому була обрана бібліотека під назвою RomeFetcher, що надавала весь необхідний функціонал.

Так як ми збираємось зберігати певні користувацькі дані, необхідно обрати для цього формат даних. Одним з найпопулярніших зараз мов розмітки користувацьких данних є XML, тож не бачу причин не обрати його, тим паче Java пропонує душе широкий спектр бібліотек для роботи з XML. Найбільш швидкою в освоєнні для мене була бібліотека JDOM, її я і обрав для подальшої роботи з XML.

Останньою бібліотекою, яку я вважаю за необхідне додати до списку, є JQS, бібліотека, що надає весь необхідний функціонал для роботи з різними соціальними мережами, використовуючи для налаштувань файли в форматі XML.

2. РЕАЛІЗАЦІЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА

2.1 Розробка GUI за допомогою SWT

SWT – кросс-платформенна бібліотека компонент для побудови графічного інтерфейсу користувача. Першим java фреймворком для побудови графічного інтерфейсу користувача була створена корпорацією Sun Microsystems бібліотека AWT (Abstract Windowing Toolkit).

Даний фреймворк використовував візуальні компоненти операційної системи. А так як вимагалося забезпечити кросс-платформенність, то виникла проблема відмінності складу візуальних компонент у різних операційних системах (ОС). З цієї причини частина корисних компонент була виключена зі складу AWT.

Для вирішення даної проблеми корпорацією Sun була почата розробка нового фреймворку. В результаті була створена дуже потужна, що включає велику кількість компонент, бібліотека Swing. На відміну від AWT компоненти Swing не залежать від компонент операційної системи. За відтворення і поведінку GUI елементів повністю відповідає бібліотека Swing. Це дозволило створити GUI компоненти, які однаково виглядають і функціонують на різних операційних системах. Для кожної операційної системи були розроблені настройки зовнішнього вигляду і поведінки, які максимально емулювали компоненти конкретної операційної системи. Використання даної технології дозволило створювати привабливі інтерфейси java додатків.

Популярність мови java привернула розробників корпорації IBM. Використання java дозволяло вирішити проблему створення безлічі версій одних і тих же продуктів для різних операційних систем. Раніше, в процесі розробки візуального інтерфейсу, була розроблена бібліотека для побудови візуального інтерфейсу CommonWidgets.

Пізніше, корпорацією IBM був ініційований проект по розробці універсальної платформи для створення java програм Eclipse. В процесі розробки Eclipse при співпраці IBM і OTI був розроблений новий графічний фреймворк, який дозволив використовувати раніше накопичений досвід без перенавчання персоналу та спрощення портування раніше створених продуктів на нову платформу. Цей фреймворк отримав назву Standard Widget Toolkit (SWT). В SWT, як і AWT максимально використовуються компоненти операційної системи. Але на відміну від AWT, відсутні в конкретній операційній системі компоненти не виключені, а емулюються. В результаті, була створена швидка високоефективна кросс-платформенна бібліотека компонент, які виглядають і поводяться як рідні для системи компоненти операційної системи. Що в свою чергу спрощує процес навчання користувачів java програм, так як SWT програми не відрізняються від звичайних додатків.

Бібліотеку SWT можна використовувати не тільки для розробки додатків на платформі Eclipse, але також і для розробки будь-яких інших додатків. При цьому можна використовувати будь-яке середовище розробки java додатків Integrated Development Enviroment (IDE), наприклад Oracle Java Developer або Borland Java Builder. Для цього досить включити необхідні бібліотеки SWT до складу java програми. Фреймворк SWT підтримує більшість популярних операційних систем. Так само існує можливість компіляції SWT java додатків в бінарний код, що підвищує продуктивність створених додатків і не вимагає встановлення java машини - Java Runtime Enviroment (JRE). Такі програми нічим не відрізняються від додатків конкретної операційної системи, а використання

сучасних IDE дозволяє швидко створювати якісні кросс-платформенні продукти для різних операційних систем.

Програмний код найпростішого програмного додатку на SWT виглядає так:

```
package ru.jugabook.swt.hello;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
public class SwtHello {
   public static void main(String[] args) {
   //Створюємо об’єкт Display для зв’язку SWT
   //з дисплеєм операційної системи
      Display display = new Display();
      //Створюємо вікно программи
      Shell shell = new Shell(display);
      shell.setText("SWT Hello");
      shell.setSize(200, 100);
      shell.open();
  //Опрацювання закриття вікна
      while (!shell.isDisposed()) {
      if (!display.readAndDispatch()) {
            display.sleep();
         }
      }
      //Ресурси операційної системи
      //повинні бути звільнені
  display.dispose(); }}
```

Також, для налагодження або запуску SWT Java програм, потрібно в редакторі «VM arguments» вказати шлях до нативної бібліотеки SWT (закладка «Arguments» панелі параметрів запуску програми). Для різних операційних систем шляхи відрізняються.

2.2 Схема проектування даних Модель-вигляд-контролер

Model-view-controller - схема використання декількох шаблонів проектування, за допомогою яких модель даних програми, користувальницький інтерфейс і взаємодія з користувачем розділені на три окремих компонента так, що модифікація одного з компонентів надає мінімальний вплив на інші. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Основна мета застосування цієї концепції полягає в поділі бізнес-логіки (моделі) від її візуалізації (подання, виду). За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах і / або з різних точок зору. Зокрема, виконуються наступні завдання:

До однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми. Не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних), для цього досить використовувати інший контролер.

Концепція MVC дозволяє розділити дані, подання та обробку дій користувача на три окремі компоненти:

- Модель (англ. Model). Модель надає знання: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна візуалізувати.
- Подання, вид (англ. View). Відповідає за відображення інформації (візуалізацію). Часто як уявлення виступає форма (вікно) з графічними елементами.
- Контролер (англ. Controller). Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель та подання для реалізації необхідної реакції.

Важливо відзначити, що як подання, так і контролер залежать від моделі. Однак модель не залежить ні від представлення, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати кілька різних подань для однієї моделі.

Для реалізації схеми Model-View-Controller використовується досить велика кількість шаблонів проектування (в залежності від складності архітектурної рішення).

Найбільш типова реалізація відокремлює вид від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій (підписка / оповіщення). При кожній зміні внутрішніх даних в моделі вона сповіщає всіх залежних від неї уявлення, і подання оновлюється. Для цього використовується шаблон «спостерігач».

При обробці реакції користувача вид вибирає, в залежності від потрібної реакції, потрібний контролер, який забезпечить ту чи іншу зв'язок з моделлю. Для цього використовується шаблон «стратегія», або замість цього може бути модифікація з використанням шаблону «команда». А для можливості однотипного поводження з подоб'єкта складно-складеного ієрархічного виду може використовуватися шаблон «компонувальник». Крім того, можуть використовуватися і інші шаблони проектування, наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного виду.

Компонентний рівень MVC в мовах програмування має ряд відступів від оригінального MVC. По-перше, відсутнє чітке розділення між представленням та контролером. По-друге, решта реалізації MVC не вимагають виділення бізнес-логіки додатка в окремий компонент - модель. Програміст самостійно приймає рішення: витрачати неабиякі зусилля по виділенню в додатку моделі або ж відразу описувати в контролері обробку даних, наприклад, звернення до бази даних.

Друге рішення призводить до проблеми проектування, для боротьби з якою і був створений шаблон MVC: якщо не виділити окрему сутність - модель, уникнути залежності бізнес-логіки від контролера неможливо. MVC не описує взаємодію моделі з даними - рівень доступу до даних розглядають інші шаблони. І все ж перераховані розбіжності з оригіналом не є порушеннями основної ідеї шаблону.

MVC задає не стільки правила поділу програми на окремі компоненти, скільки правила їх взаємодії. У той час як уявлення і контролер залежать від моделі, модель не залежить ні від представлення, ні від контролера. Це ключова

особливість поділу, яка дозволяє працювати з моделлю, а значить, і з бізнес-логікою програми, незалежно від візуального представлення.

Крім підтримки реалізації MVC з боку стандартного комплекту засобів розробки мов програмування, існує досить багато програмних платформ, які пропонують готові рішення по реалізації програм на основі даного шаблону.

Подання безпосередньо управляє відображенням інформації користувачеві. Сама ця інформація в термінах шаблону проектування MVC називається моделлю. Контролер співпрацює з поданням, гарантуючи інтерпретацію дій користувача над даними, знову ж таки - відображенням моделі. Тому уявлення і контролер можуть бути обмежені типом даних (моделлю), з яким вони працюють. Це обмеження дозволить використовувати уявлення і контролер спільно для роботи з одним типом даних - моделлю.

Навпаки, модель не повинна мати обмежень. Обмеження на тип об'єкта, який може функціонувати в якості моделі, в свою чергу обмежить рамки використання тріади MVC. Необхідно, щоб будь-який об'єкт міг бути моделлю. Наприклад, число з плаваючою точкою може виступити моделлю для відображення температури, а рядок - моделлю текстового редактора.

З іншого боку, модель може бути обмежена типом даних, який вона представляє, і який успадковується від суперкласу для всіх можливих типів - класу Object. Тип даних, яким оперує модель, назвемо властивістю моделі.

В деяких описах оригінальної реалізації MVC згадується про пасивні та активні моделі. Пасивна модель не обізнана про існування подання, контролера, і навіть про свою участь в MVC-тріаді. Контролер відстежує зміни моделі і оповіщає подання. При цьому або контролер передає поданням інформацію про зміни, або подання самостійно вибирає дані з моделі.

2.3 Бібліотека RomeFetcher

ROME відтворює стрічки новин (RSS і Atom), як екземпляри com.sun.syndication.synd.SyndFeed інтерфейсу. SyndFeed інтерфейс і його властивості слідують моделі Java Bean. Всі стандартні реалізації ROME являють собою класи невеликого розміру.

ROME включає аналізатори для обробки стрічок та перетворення їх в SyndFeed об'єкти. Клас SyndFeedInput обирає парсери керуючись типом поточної стрічки новин. Розробнику не потрібно турбуватися про вибір правильного парсеру для обробки RSS стрічки, SyndFeedInput подбає про неї, перевіривши її внутрішню структуру. Все, що необхідно, щоб зчитати новинну стрічку використовуючи ROME, це використати два наступні рядки коду:

SyndFeedInput input = new SyndFeedInput();

SyndFeed feed = input.build(new XmlReader(feedUrl));

У першому рядку створюється екземпляр *SyndFeedInput,* який працюватиме з будь-яким типом стрічок (RSS і Atom версії). Другий рядок вказує *SyndFeedInput* зчитати новинну стрічку із символьного потоку, отриманого за адресою, що вказує на стрічку новин. *SyndFeedInput.build ()* повертає об'єкт типу *SyndFeed*, який може бути легко оброблений

Нижче наведено повний код для додатків Java, який читає новинну стрічку і виводить SyndFeed bean до консолі:

```
package com.sun.syndication.samples;
import java.net.URL;
import java.io.InputStreamReader;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.io.SyndFeedInput;
import com.sun.syndication.io.XmlReader;
```

```
// Читує і роздруковує будь-якку RSS/Atom стрічку.
public class FeedReader {
    public static void main(String[] args) {
        boolean ok = false;
        if (args.length==1) {
            try {
                URL feedUrl = new URL(args[0]);
                SyndFeedInput input = new SyndFeedInput();
                SyndFeed feed = input.build(new XmlReader(feedUrl));
                System.out.println(feed);
                ok = true;
            }
            catch (Exception ex) {
                ex.printStackTrace();
                System.out.println("ERROR: "+ex.getMessage());
            }
        }
        if (!ok) {
            System.out.println();
            System.out.println("FeedReader reads and prints any RSS/Atom feed type.");
            System.out.println("The first parameter must be the URL of the feed to read.");
            System.out.println();
        }}}
```

2.4 Бібліотека JDOM

JDOM це вільна реалізація Java-DOM для XML, створена з урахуванням особливостей мови і платформи Java. JDOM інтегрується з Document Object Model (DOM) і Simple API для XML (SAX), підтримує XPath і XSLT. В JDOM використовуються зовнішні парсери для генерації документів. JDOM розроблявся Джейсоном Хантером і Бреттом МакЛофліном, починаючи з березня 2000 року. Він є частиною Java Community Process. Назва розшифровується як JDOM Java Document Object Model.

Нехай файл "foo.xml" містить наступний XML-документ:

```
<shop name="shop for geeks" location="Tokyo, Japan">
  <computer name="iBook" price="1200$" />
  <comic_book name="Dragon Ball vol 1" price="9$" />
  <geekyness_of_shop price="priceless" />
</shop>
```

Наступний приклад коду виробляє розбір XML-файла в дерево Java-об'єктів за допомогою JDOM:

```
SAXBuilder builder = new SAXBuilder();
Document doc = builder.build(new FileInputStream("foo.xml"));
Element root = doc.getRootElement();
// root.getName() is "shop"
// root.getAttributeValue("name") is "shop for geeks"
// root.getAttributeValue("location") is "Tokyo, Japan"
// root.getChildren() is a java.util.List object that contains 3 Element objects.
```

DOM можна створити не тільки з файлу або потоку, а й з простих об'єктів-елементів.

Кореневий елемент = новий елемент ("магазин") / / тут <shop> </ магазин> є коренем

Документ Doc = новий документ (корінь);

Так можна побудувати дерево з об'єктів-елементів та згенерувати з нього XML-файл:

Element root = new Element("shop");

Document doc = new Document(root);

Так можна побудувати дерево з об'єктів-елементів та згенерувати з нього XML-файл:

```
Element root = new Element("shop");
root.setAttribute("name", "shop for geeks");
root.setAttribute("location", "Tokyo, Japan");
Element item1 = new Element("computer");
item1.setAttribute("name", "iBook");
item1.setAttribute("price", "1200$");
root.addContent(item1);
// те ж саме для інших елементів
XMLOutputter outputter = new XMLOutputter();
outputter.output(new Document(root), new FileOutputStream ("foo2.xml"));
```

2.5 Середовище розробки Eclipse IDE

Спочатку Eclipse розроблялася фірмою IBM як наступник середовища розробки IBM VisualAge, в якості корпоративного стандарту IDE для розробки на різних мовах під платформи IBM. За відомостями IBM, проектування та розробка коштували 40 мільйонів доларів. Вихідний код був повністю відкритий і зроблений доступним після того, як Eclipse був переданий для подальшого розвитку незалежного від IBM спільноті.

В Eclipse 3.0 (2003 рік) були обрані специфікації сервісної платформи OSGi, як архітектура часу виконання. З версії 3.0 Eclipse перестав бути монолітною IDE, яка підтримує розширення, а сам став набором розширень. В основі лежать фреймворк OSGi і SWT / JFace, на основі яких розроблений наступний шар - RCP (Rich Client Platform, платформа для розробки повноцінних клієнтських додатків). RCP є основою не тільки для Eclipse, але і для інших RCP-додатків, наприклад Azureus і File Arranger. Наступний шар - сам Eclipse, що представляє собою набір розширень RCP - редактори, панелі, перспективи, модуль CVS і модуль Java Development Tools (JDT).

З 2006 року фонд Eclipse координує щорічний загальний реліз , який відбувається в червні. Кожен випуск містить у собі платформу Eclipse, а також ряд інших проектів Eclipse.

Eclipse служить в першу чергу платформою для розробки розширень, чим він і завоював популярність: будь-який розробник може розширити Eclipse своїми модулями. Вже існують Java Development Tools (JDT), C / C + + Development Tools (CDT), розроблювані інженерами QNX разом із IBM, і кошти для мов Ada (GNATbench, Hibachi), COBOL, FORTRAN, PHP та ін від різних розробників. Безліч розширень доповнює середу Eclipse менеджерами

для роботи з базами даних, серверами додатків і ін

Eclipse JDT (Java Development Tools) - найбільш відомий модуль, націлений на групову розробку: середа інтегрована з системами управління версіями - CVS, GIT в основний постачання, для інших систем (наприклад, Subversion, MS SourceSafe) існують плагіни. Також пропонує підтримку зв'язку між IDE і системою управління завданнями (помилками). В основній поставці включена підтримка трекера помилок Bugzilla, також є безліч розширень для підтримки інших трекерів (Trac, Jira та ін.) В силу безкоштовності і високої якості, Eclipse в багатьох організаціях є корпоративним стандартом для розробки додатків.

Eclipse написана на Java, тому є платформо-незалежним продуктом, за винятком бібліотеки SWT, яка розробляється для всіх поширених платформ (див. нижче). Бібліотека SWT використовується замість стандартної для Java бібліотеки Swing. Вона повністю спирається на нижележащую платформу (операційну систему), що забезпечує швидкість і натуральний зовнішній вигляд користувальницького інтерфейсу, але іноді викликає на різних платформах проблеми сумісності та стійкості додатків.

3. ОСОБЛИВОСТІ СТВОРЕНОГО ДОДАТКУ

Першим кроком для створення програмного додатку було встановлення та налаштування усіх необхідних програмних компонентів. Для розробки було використано IDE Microsoft Visual Studio 2008 з використанням бібліотек ADO.NET та спеціалізованого додатку для створення звітів у форматі Crystal Reports.

Так, як зручного інтегрованого функціоналу для створення звітів в .NET не передбачено, нам було видано завдання встановити, налаштувати та використовувати бібліотеки Crystal Reports.

Crystal Reports XI розширює можливості генерації звітів Crystal Reports на платформу Microsoft .NET. Ми використовували Crystal Reports Designer в Visual Studio .NET (VS .NET) для створення нового звіту Crystal.

Ми зберігали звіт на локальному комп'ютері і працювали з ним за допомогою Windows Forms Viewer.

При додаванні звітів в додаток .NET за умовчанням вони додаються до додатку як вбудований ресурс. Це означає, що звіт буде скомпільований в маніфесті збірки, і не завантажуватиметься з окремого файлу звіту (.RPT).

Робота із звітом, як вбудованим ресурсом дозволяє поставляти і розгортати додатки .NET без окремого постачання файлів звітів. Перевага полягає у відсутності для додатку необхідності в розгортанні зовнішніх файлів звітів, тому кінцеві користувачі не можуть їх модифікувати. Але є і недолік - якщо звіт необхідно модифікувати, доведеться перекомпілювати весь програмний додаток і знову розвернути його для збереження змін в звіті. Для запобігання компіляції звітів в маніфест збірки, як вбудованого ресурсу ми значення поля Build Action з Embedded Resource на None.

При використанні вищезгаданого кроку звіт не буде скомпільований в маніфест збірки, таким чином, звіт завантажуватиметься з жорсткого диска. Звіти можуть бути завантажені з диска за допомогою вказівки файлового шляху, з використанням методу Load об'єкту Reportdocument.

При розгортанні додатків, що не мають вбудованих файлів звітів, ці файли повинні бути вручну додані при настройці проекту. Перевага зберігання звітів поза збіркою полягає в тому, що звіти можуть бути легко модифіковані і повторно розгорнені без необхідності в перекомпіляції і розгортанні всієї збірки. Недолік полягає в тому, що жорстко задані типи об'єктів не можуть бути використані в додатку, а звіти повинні завантажуватися з жорсткого диска.

Існує два методи розгортання runtime-файлів Crystal Reports XI на цільовому комп'ютері:

- Використання модулів злиття (merge modules)
- Використання інтерактивної установки (також відомою як "необмеженою" (redistributable) або серверної установки)

У обох сценаріях встановлюються одні і ті ж файли, проте, методи розгортання відрізняються. Модулі злиття призначені тим розробникам, які хочуть включити runtime-файли Crystal Reports разом з пакетом в подальше застосування. Інтерактивна установка призначена для тих, хто хоче просто розвернути ці файли на серверному комп'ютері як автономне завдання.

Для виконання поставленої задачі нам потрібно було вказати особливі параметри установки, тож необхідно було застосувати перший варіант розгортання. Компоненти, призначені для додавання в настройку проектів для постачання і розгортання додатків Crystal Reports 10, включають:

- Файли звітів (.RPT)
- Модулі злиття Crystal Reports

Файли звітів (.RPT) слід поставляти при розгортанні додатків .NET. Спосіб розгортання файлів звітів залежить від наявних потреб і архітектури додатку.

Модулі злиття, які необхідно додавати в настройку проекту, встановлюється з Crystal Reports XI в каталог «C:\Program Files\Common Files\merge\Modules\».

Попередні версії Crystal Reports для розгортання звітів вимагали включення трьох або більше модулів в налаштування проекту. Для Crystal Reports XI, необхідно включити тільки один файл злиття, при цьому другий є опціональним: Crystal11_net_embeddedreporting.msm

Crystalreports11_maps.msm (опціонально)

Перший модуль злиття містить елементи управління Crystal Report Viewer, SDK-сборки Crystal Reports .NET, SDK-сборки RAS, ядро обробки звітів, компоненти бази даних і компоненти експорту. Цей модуль злиття являється необхідним для додатків Crystal Reports XI .NET.

Ми запустили на виконання файл модуля злиття (.MSI) і при появі запрошення ввели ключовий код Crystal Reports XI, який нам було надано на робочому місці. Другий модуль злиття використовується для інсталяції файлів, потрібних для відображення в звітах географічних карт. Цей модуль злиття потрібний тільки в тому випадку, якщо в звіт включена картографічна інформація. Тому його розгортання ми не виконували.

У Crystal Reports 11 і в Visual Studio .NET, модуль злиття Crystal11_net_embeddedreporting.msm дозволяє вам вказувати опції для драйверів баз даних, драйверів експорту і віртуального IIS-каталога crystalreportviewers11. За умовчанням, все це не встановлюється з Crystal11_net_embeddedreporting.msm, тому їх довелося вказувати окремо.

Було встановлено наступні драйвера баз даних:

- Installcrdb_adoplus - драйвер бази даних Crystal Reports для Microsoft ADO.NET.
- Installcrdb_cdo - дайвер бази даних Crystal Reports для об'єктів даних Crystal.
- Installcrdb_com - драйвер бази даних Crystal Reports для провайдера даних COM.
- Installcrdb_dataset - драйвер бази даних Crystal Reports для провайдера Dataset.
- Installcrdb_odbc - драйвер бази даних Crystal Reports для ODBC.
- Installcrdb_oracle - драйвер бази даних Crystal Reports для Oracle.
- Installcrdb_query - драйвер бази даних Crystal Reports для даних запитів.

Було встановлено наступні драйвера експорту:

- Installu2odbc - драйвер експорту Crystal Reports для формату експорту ODBC.
- Installu2fpdf - драйвер експорту Crystal Reports для експорту у форматі PDF.
- Installu2frec - драйвер експорту Crystal Reports для формату експорту у вигляді записів (Records).

- Installu2frtf - драйвер експорту Crystal Reports для формату експорту Rich Text (rtf).
- Installu2sepv - драйвер експорту Crystal Reports для формату значень з роздільниками.
- Installu2ftext - драйвер експорту Crystal Reports для текстового формату експорту.
- Installu2fxls - драйвер експорту Crystal Reports для формату експорту Excel.
- Installu2fxml - драйвер експорту Crystal Reports для формату експорту XML.

Оскільки проекти, створені в VS.NET відносяться до .NET Framework, то Framework також повинен поставлятися і встановлюватися на цільовому комп'ютері. Коли виконується настройка проекту, зазвичай Framework автоматично не включається – це ми робили власноруч.

Коли який-небудь з файлів звітів використовує об'єкти баз даних ADO .NET, потрібно буде включити наступні два об'єднані модулі:

Vc_user_crt71_rtl_x86_---.msm і Vc_user_stl71_rtl_x86_--- .msm. Ці два об'єднані модулі потрібні тому, що драйвер бази даних, Crdb_adoplus.dll, є залежним від наступних файлів, які встановлюють об'єднані модулі:

- Msvcr70.dll
- Msvcr71.dll

Наступним етапом роботи стала розробка програмного коду додатку для роботи з базою даних ГІОЦ УЗ, а саме з наданими для тестування програми таблицями. Для початку потрібно було встановити з’єднання з Oracle 10g. Для цього ми використовували минулорічні напрацювання, дещо їх модифікувавши до теперішнього контексту. Було розроблено клас OracleFactory:

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OracleClient;
using System.Linq;
using Core.Extensions;

namespace UZ_BBPE.Core
{
  class OracleFactory : IDisposable
  {
    #region Variables
    private bool _disposed = false;
    private String _connectionString = "Data Source="";Password="";User ID="""";
    private OracleConnection _connection;
    private String _lastCommandName;
    #endregion

    #region Constructor
```

```
public OracleFactory()
 {
  InitializeConnection();
 }
 public OracleFactory(String connectionString)
 {
  _connectionString = connectionString;
  InitializeConnection();
 }
 private void InitializeConnection()
 {
  Commands = new Dictionary<String, OracleCommand>();
  RollbackCommandNames = new List<string>();
  try
  {
   _connection = new OracleConnection(_connectionString);
   _connection.Open();
  }
  catch (OracleException oraEx)
  {
   Console.WriteLine("Connection failed with message: " + oraEx.ToString());
  }
 }
 #endregion
```

Для забезпечення цілісності даних та збереження зроблених змін до бази було розроблено власний метод Dispose в поєднанні з використаним інтерфейсом IDisposable, що в теорії має забезпечити збереження даних поточної транзакції при виникненні критичних ситуацій:

```
#region IDisposable
  ~OracleFactory()
  {
   Dispose(false);
  }
  public void Dispose()
  {
   Dispose(true);
   GC.SuppressFinalize(this);
  }
  protected virtual void Dispose(bool disposeManagedResources)
  {
   if (!_disposed)
   {
    foreach (String commandName in Commands.Keys)
    {
     if (RollbackCommandNames.Contains(commandName))
      Commands[commandName].Transaction.Rollback();
     else
      Commands[commandName].Transaction.Commit();
```

```
  }
      if (disposeManagedResources)
      {
        foreach (String commandName in Commands.Keys)
        {
          Commands[commandName].Dispose();
        }
        //Dispose connection information
        _connection.Close();
        _connection.Dispose();
      }
      _disposed = true;
    }
  }
  #endregion
```

Для отримання даних з бази було використано пару методів, в залежності від специфічності запиту і отриманні бажаних даних. Найпростіший запит має вигляд:

```
string NameUser;
string Pass;
string sCommString = "SELECT * FROM ASSSAD.AACCESS WHERE N_USER='"
+ NameUser + "' and PASSWORD='" + Pass + "'";
orclComm.CommandText = sCommString;
orclComm.Connection.Open();
```

[underline]int i = orclCommAssad.ExecuteNonQuery();[/underline]

orclCommAssad.Connection.Close();

Користуючись такого типу методами було створено частину додатку, яка мала на меті керування даними таблиці.

ВИСНОВКИ

Під час написання даної дипломної роботу було розглянуто основні принципи та засоби створення веб-додатків. Були досліджені існуючі лабораторії функціонально-логічного моделювання та проведено їх порівняння.

Була розроблена власна лабораторія функціонально-логічного моделювання у вигляді клієнт-серверного веб-додатка. Для клієнтської частини було обрано Javascript, для серверної - Java, як основна серверна мова. Для взаємодії клієнта та сервера було застосовано Ajax підхід.

Отримана реалізація була протестована на різних вхідних описах апаратури і видавала ті ж результати, що і аналоги. Тому можна зробити висновки, що дана реалізація працює справно.

У подальшому слід зосередити свою увагу на збільшенні та покращенні функціоналу даного веб-додатка. Отримана програмна реалізація може бути застосована для функціонально-логічного моделювання. Також дана розробка може бути імплементована в певний начальний сервіс для полегшення входження студентів в моделювання та для демонстрації роботи певних модулів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Блинов, И.Н. Java/ Промышленное программирование: практ.пособие/И.Н. Блинов, В.С. Романчик.
2. Офіційний сайт Icarus Verilog - Режим доступа: http://iverilog.icarus.com/ - дата доступу: 05.2020.
3. Анопрієнко О. Я. Методика проектування Web-додатків з використанням платформи Java EE/Анопрієнко О.Я.// Інформатика і комп'ютерні технології - 2010 - №VI - С. 160-165.
4. Хорстманн, Кей С., Корнелл Гари. Java2. Основы
5. Budi Kurniawan. Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions / Budi Kurniawan: New Riders Publishing, April 2018 - 976p.
6. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. - П. : "Питер", 2015. - 656 с.
7. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML and CSS Bible, 5th Edition. - М.: "Диалектика", 2010. - 656с
8. Дейв Крейн, Бер Бибо, Джордон Сонневельд. Ajax in Practice.